Boston Transportation Analysis: Fastest Route to Northeastern

Charlie Deane and Peter Li

December 15, 2022

Abstract

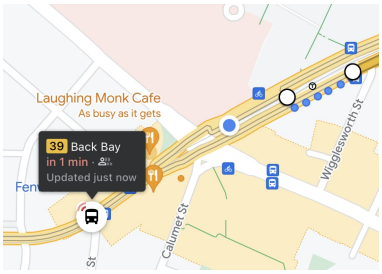
With over 650,000 residents, 400,000 of whom commute on a daily basis, the community of Boston exerts undue pressure on the city's public transit infrastructure. Due in part by the culmination of under-investment, and increasing population, the transportation network has become unreliable for residents and students. We attempt to perform a regression analysis on the arrival times and headway's of Boston's public transportation system using the dataset provided by the Massachusetts Bay Transportation Authority (MBTA) [Geo19; Geo20a; Geo20b] and local weather data [Bro21]. Modeling and predicting arrival times and headway's using a combination of regression models and deep neural networks, and different feature encoders.

1 Introduction & Motivation

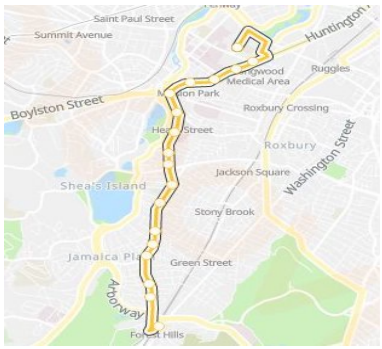
Historically, the authors of this paper have been unpunctual when attending their Data Mining & Machine Learning course. While this is in part due to their lackluster sleep schedule, some blame can be put on the irregularity of the Boston transportation system. With this motivating reason, we outline a paper to analyze Boston's bus system to maximize the likelihood of the authors' timely arrival to class.

There exist many modern tools to estimate travel times like Google Maps, Apple Maps, and the MBTA API, which are generally accurate for roadway congestion. However, these applications can be especially unreliable when estimating the public transit arrivals and headways. Public transportation is vital in densely populated cities like Boston and is especially important for the faculty and student population at Northeastern University.

Improving the accuracy of bus arrival time predictions may have significant benefits for the efficiency of the transportation system in Boston. Accurate arrival time predictions can help passengers better plan their trips and reduce waiting times at bus stops, leading to a more convenient and reliable commuting experience for students and citizens. In addition, more accurate arrival time predictions can also help the transportation agency optimize the deployment of its resources, potentially leading to cost savings and reduced greenhouse gas emissions. Overall, this research project has the potential to make a meaningful impact on the transportation system in Boston.



(a) Google Maps



(b) Google Maps

Figure 1.1: Google Maps Data

1.1 Existing Literature

To the best of our knowledge the public transit delays in the city of Boston have not been significantly studied or have their information publicly available. Most likely this information is unhelpful in the data and age of GPS technology. Bus locations can be tracked in real time (see Figure 1.1).

There exists previous work which model the train delays in the city of Stockholm, Sweden using Neural Networks and decision trees; however, this work [NH] is attached to the specific location of the transit system, and it's findings are unique to the operations and weather of the Swedish city. Additionally, the study used fundamental techniques (low depth Neural Networks) to draw its conclusions.

1.2 Methodology

Our analysis focuses specifically on the bus arrival and departure at bus stop locations near the Northeastern University Boston campus (e.g. the inbound route 39). We formulate our model by extracting the features from weather data, the scheduled location (stop location), and the scheduled time of arrival of the bus:

$$\Gamma(x) = (W_{weather}, L_{location}, T_{time})$$

We model our features using 3 feature encoding methods: (1) simple encoding of time, where each year, month, day, hour, etc. is fed directly as a feature to the model (2) binary encoding of time and location, where each year, month, day, etc. is encoded as a bitmap before feeding it into the neural network and (3) positional encoding [Mil+20] where is time is mapped to a higher dimensional space using high frequency functions:

$$\gamma(t) = (\sin(2^0\pi t), \cos(2^0\pi t), ..., sin(2^{L-1}\pi t), \cos(2^{L-1}\pi t))$$

The function γ is applied to each time component (year, month, day, etc.) where we set $L = 6$ in our experiments as a constructional hyper-parameter. Positional encoding has been used in previous work for transformer architecture and even neural radiance fields. However, it has not been applied to time variant data in this context. Our basic regression models include: vanilla ordinary least squares, lasso regression, and ridge regression models (see Figure 1.2).

Model	Form	Problem Function
Vanilla Linear Regression (OLS)	$(X'X)^{-1}X'Y$	$min_{\theta \in \mathbf{R}} \sum_{i=1}^{N}(y_i - \theta^T x_i)^2$
Ridge Regression	$(X'X + \lambda I)^{-1}X'Y$	$min_{\theta \in \mathbf{R}} \sum_{i=1}^{N}(y_i - \theta^T x_i)^2 + \lambda \sum_{j=1}^{P} \theta_j^2$
Lasso Regression	$sgn(\hat{\theta}^{LS})(\|\hat{\theta}^{LS}\| - \alpha)^+$	$min_{\theta \in \mathbf{R}} \sum_{i=1}^{N}(y_i - \theta^T x_i)^2 + \lambda \sum_{j=1}^{P} \|\theta_j\|$

Figure 1.2: Basic Regression Models

We compare three neural network models. Our first model has 2 linear layers (simply an input and an output layer) using Sigmoid activation functions and Mean Squared Error as our loss function. Our second neural network uses 4 linear layers with a hidden layer width of 10 neurons, this time using ReLU as our activation function (see Figure 1.3). Our final model uses 4 linear layers (each channel down-sampling by 2), using ReLU as our activation function, but with a dropout probability of 0.2 applied before each linear layer. We use the Stochastic Gradient Descent Optimizer in PyTorch without momentum.

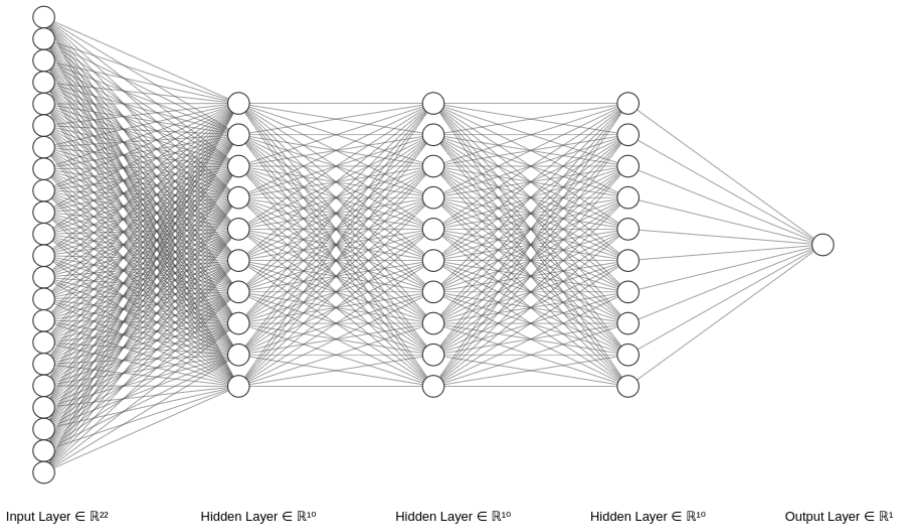


Figure 1.3: Four-Layer Neural Network (Simple Encoding)

2 Technical Approach

We sourced Boston's daily temperature range (measured in degrees Fahrenheit), and daily precipitation levels (measured in inches) of the year 2020 from Extreme Weather Watch [Bro21]. This data-set amounted to 365 rows, representing 1 sample for each

day in the year. We also utilized the 2020 Bus Arrival and Departure times from the MBTA, which included key metrics pertaining to stop location, route dates, scheduled arrival times, and actual arrival times. We choose this set, as it represents the most recent information published by the MBTA, giving the most representative picture to the population size, and transit infrastructure status.

Day	High (°F)	Low (°F)	Precip. (inches)	Snow (inches)
January 1	43	36	0.00	0.0
January 2	49	34	0.00	0.0
January 3	52	44	0.00	0.0

(a) Boston Weather Data

service_date	route_id	direction_id	stop_id	start_time_sec	end_time_sec	headway_time_sec	destination	ObjectId
2020-01-01 05:00:00+00:00	Blue	0	70049	27258	28042	784	Bowdoin	1
2020-01-01 05:00:00+00:00	Blue	0	70049	39944	40555	611	Bowdoin	2
2020-01-01 05:00:00+00:00	Blue	0	70049	42466	43058	592	Bowdoin	3

(b) Boston Route Data

Figure 2.1: MBTA and Extreme Weather Watch Datasets

To adequately leverage these two sets of information, required a multifaceted approach when it came to filtering and massaging the data. Prior to ingesting the CSV files, each dataset, representing 3 months of bus activity, occupied between 742 MB, and 891 MB. Since we were constrained to limited processing capabilities, required us to chunk data into subsets of 1000 rows, and iteratively filtering for rows with null records that exceeded (25) percent of the total columns. As previously outlined in introduction, our main focus is on Northeastern centric arrivals, therefore, we filtered for Route 39, (represented as "39") in the "route_id" field and stratify samples from each of the quarterly datasets (p=0.002). All in, this reduced roughly 24,980,498 rows down to 1,234 rows.

We proceeded to adjust the data, still in its native form, into usable type values to be ingested as features in our various models. This involved modifying the "service_date" to a date object, instead of a datetime string, converting the "scheduled" and "actual" column to the total number of seconds after midnight, and appending the daily weather samples to the corresponding date.

The previous year's quarter-over-quarter (QoQ) change was no more than 15 megabytes, raising doubts about the cause of such variance in the 2020 dataset. In simpler terms, this difference can be attributed to the decrease in bus operations between March and June due to the COVID-19 Pandemic. On June 22^{nd}, bus frequency was restored [WBU], although it remained below 2019 levels.

To generate our features, we took each row of the MBTA processed dataset and mapped the corresponding weather measurements recorded on the same day as the bus event. We calculated the labels, time delta denote δ, by subtracting the *Scheduled* and *Actual* arrival times found in the dataset. We proceeded to use our processed datasets to split into Test (80%), Train (10%), and Validation (10%) datasets.

2.1 Training Our Model

Our regression models were developed using the Scikit-Learn framework, and our neural network models utilized the PyTorch neural network modules. We trained our models for ~ 4 hours on the Northeastern Discovery Cluster using an a100 GPU and an AMD EPYC 7543 (2.80GHz) CPU running on CUDA 11.3.

2.2 Regression Models

We evaluated three regression models for each of our encoders. The performance of our regression models exceeded our expectations. Initially, we hypothesized that the linear regression models would perform poorly due to the absence of an obvious correlation between our features and the time δ. However, our experiments demonstrated there was at least some encoded feature information that maps to the time deltas. Furthermore, we expected that our vanilla linear regression model to perform the worst among the three models due to its lack of a regularization term. However, that did not prove to be the class. Our lasso regression model proved to perform the worst with a difference of about 0.05 R^2 score between it and the other two models. This could likely denote the non-collinearity between our features, a property well-suited for lasso regression.

When analyzing our weights, we found the absolute magnitude of the $time\_point\_order$ weights greatly affected the accuracy of the model. More importantly, we noted that buses earlier in the time point order (closer to the site of origin) had a higher likelihood of the bus arriving on time. Additionally, we found that the hour of the day significantly affected the timely arrival of transport. With 5 PM being the time of day with the worst delays for commuters and 6 PM, 11 PM, and 12 PM being the most accurate for scheduled and arrival times (lowest time δ).

	Simple (MSE)	Binary (MSE)	Positional (MSE)
OLS	29902.91	30849.09	30850.44
Ridge	29903.42	30720.25	30842.13
Lasso	29931.69	30030.44	30046.64

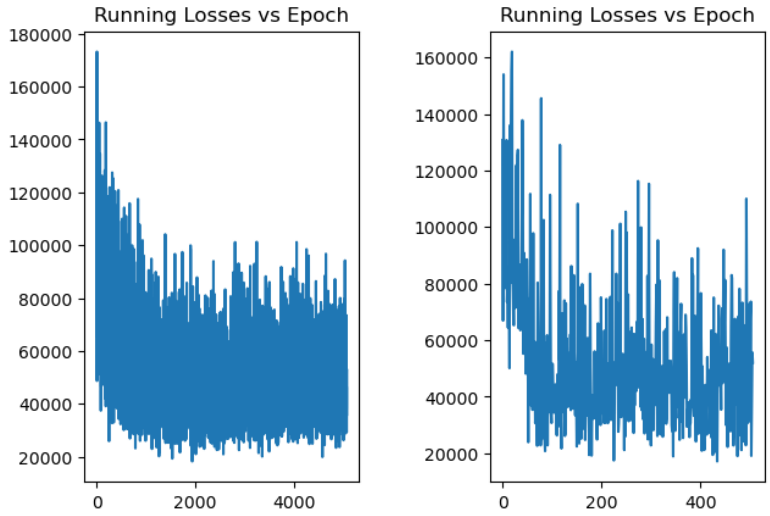
Figure 2.2: Regression Models Mean Squared Error (seconds^2)

	Simple (MAE)	Binary (MAE)	Positional (MAE)
OLS	125.67	129.88	129.87
Ridge	125.68	129.44	129.85
Lasso	125.73	126.74	126.59

Figure 2.3: Regression Models Mean Absolute Error (seconds)

2.3 Deep Neural Networks

Of our 4 models, the 4-layer neural network (on average) seemed to perform the best. Although it is quite complex to explain neural network outcomes, we assume that this



(a) Example DNN Running Loss for 4-Layer Positional Encoding

(b) Example DNN Running Loss for 4-Layer Simple Encoding

Figure 2.5: Model Losses

is because the down sampling approach we took with the 4-layer network. Instead of immediately scaling from the high feature count of 2000 or 4000 in binary and positional encoding to a layer of width 10 (as we did with the 2 layer approach). Our 4-layer network scaled the features down to 256 in the first layer and gradually decreased each layer size. It is likely that due to sudden layer down-sampling of the 2-layer network, significant information was lost, leading to the lackluster performance. Nonetheless, our neural network models did not match our expectations. We assumed that our neural network models would match or perform better than linear regression; however, that was not the case. The neural network had performed slightly poorer than our regression models. This fact, we might be able to attribute to the sudden down-sampling of the input layer. We hypothesize that if we increased the size of the initial hidden layer, our model might perform better. On the other-hand, our model did perform respectably, with an average loss of near 200 seconds.

Our dropout model also proved to perform worse than expected. This is likely due to the fact that some of our features are only encoded once and our model is unlikely to over-fit our data. Nonetheless, it does perform comparably to the results of the other neural network models.

	Simple	Binary	Positional
2-Layer	40897.5312	34795.3438	36828.5078
4-Layer	34797.4531	34774.1758	34791.3828
Dropout	41986.8438	35098.0898	37920.9102

Figure 2.4: Deep Neural Network Test Loss

We tested various loss functions (ReLU and Sigmoid) but did not observe a signif-

icant difference in performance between each function. We also adjusted other hyper-parameters in our validation of our model (learning rate, batch size, epochs, and hidden dimension size). After training our models and tweaking specific parameters, we ultimately found that a learning rate of 1e-5, and a batch size of 50 was optimal.

3 Encoders Discussion

Comparing the three positional encoders aforementioned our results show a significant performance boost from the binary encoding of our time point features. We hypothesize that this encoding maps the simple encoding into discrete bins. Making it easier for our models to associate specific weights to time points. For example, if 5 PM had heavy delays which ended at 6 PM, but then started back up at 7 PM, the linear regression model would have a difficult time associating a specific weight to the hour attribute. Moreover, since time wraps around (hour 0 is hour 24) regression models would have a difficult time associating a correlation.

4 Summary and Discussions

The objective of this paper investigated the potential to accurately predict travel time between student's dormitories and the Northeastern University Boston campus. Using a series of differentiated models, we found our binary coder significantly outperformed our simple encoder. Further analysis revealed this contrast can be attributed to the representation of our data set, expressed in months/day of the month which, in its native plain-text encoded state, would indicate a 364 day delta from December 31st to January 1st rather than a 1 day delta. With this in mind, we found the vanilla and ridge linear regression model's slightly outperformed the 2 and 4 hidden layer deep neural networks. We assume our DNN's under performed due to the lack of neurons in the first hidden layer. If we were to increase the width of our first hidden layer, we assume that the loss would decrease. However, we should note that this would also demand exponentially more compute resources and time.

4.1 Comparison and Future Work

In the status quo, Google Maps' public transit routes offer limited information to users, particularly in the late evening hours. After 10:00 PM, Google Maps stops providing estimated arrival times for buses, even though these routes continue to operate for an additional 2 to 3 hours. This can be problematic for users who rely on Google Maps to plan their trips and need to know when the bus is expected to arrive at their stop. It is also worth noting that Google Maps' estimated arrival times are not always accurate. This can cause confusion and inconvenience for users who are relying on the app to plan their public transit transportation. Regarding future work, we would like to expand this model to additional routes in the surrounding area, as to offer a more effective method of predictable public transit. Given the importance of public mass transit, and the increasing reliable on these resources, we would ultimately like to create an interactive graphical representation of our model. Additionally, we would like to calculate actual travel route times between stops using a scheduling optimizer. We should also note

that above ground transit is affected by street traffic, so an overlay with the Google Maps API would be beneficial in constructing more precise models.

References

[Geo19] Massachusetts Geodot. "MBTA Bus Arrival Departure Times 2019". In: (2019).

[Geo20a] Massachusetts Geodot. "MBTA Bus Arrival Departure Times 2020". In: (2020).

[Geo20b] Massachusetts Geodot. "MBTA Rapid Transit Headways 2020". In: (2020).

[Mil+20] Ben Mildenhall et al. *NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis*. 2020. DOI: 10.48550/ARXIV.2003.08934. URL: https://arxiv.org/abs/2003.08934.

[Bro21] H Brothers Inc. *Boston Weather in 2020*. 2021.

[NH] Robert Nilsson and Kim Henning. *Predictions of train delays using machine learning Förutsägelser av tågförseningar med hjälp av maskininlärning*. https://www.diva-portal.org/smash/get/diva2:1217917/FULLTEXT01.pdf. Accessed: 2022-12-16.

[WBU] WBUR. *Increased T Service Less Than Two Weeks Away*. en. https://www.wbur.org/news/2020/06/10/mbta-service-increase-coronavirus-reopening-phase-two.. Accessed: 2022-12-16.

Program Repository found here